CIRRELT

Centre interuniversitaire de recherche
sur les réseaux d'entreprise, la logistique et le transport

Interuniversity Research Centre
on Enterprise Networks, Logistics and Transportation

On Storage Assignment Policies for Unit-Load Automated Storage and Retrieval Systems

Jean-Philippe Gagliardi
Jacques Renaud
Angel Ruiz

June 2010

CIRRELT-2010-25

Bureaux de Montréal :
Université de Montréal
C.P. 6128, succ. Centre-ville
Montréal (Québec)
Canada H3C 3J7
Téléphone : 514 343-7575
Télécopie : 514 343-7121

Bureaux de Québec :
Université Laval
2325, de la Terrasse, bureau 2642
Québec (Québec)
Canada G1V 0A6
Téléphone : 418 656-2073
Télécopie : 418 656-2624

www.cirrelt.ca

On Storage Assignment Policies for Unit-Load Automated Storage and Retrieval Systems

Jean-Philippe Gagliardi, Jacques Renaud*, Angel Ruiz

Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), and Department of Operations and Decision Systems, 2325, de la Terrasse, Université Laval, Québec (Québec), G1V 0A6

Abstract. This paper focuses on the management of *Automated Storage and Retrieval Systems* (AS/RS) from a practical standpoint. Most of articles in the recent AS/RS literature rely on analytical models that are based on tight assumptions. However, some of these assumptions are not clear cut and therefore the validity of the results reported in the theoretical works need to be clearly assessed when applying them to a particular industrial setting. To this end, this paper presents a discrete-event simulator that was designed to accurately reproduce the characteristics of an industrial AS/RS owned by a large manufacturer in the food industry located in Quebec City, Canada. Various scenarios inspired by real data provided by our industrial partner were used to compare the system performance under several storage assignment policies. Our experimental results confirm our hypothesis that the system behavior deviates from the theoretical expectations as soon as the most simplistic, yet realistic conditions are considered.

Keywords. Automated storage and retrieval systems, storage assignment, class-based storage, full turnover-based storage, discrete-event simulation, warehousing.

Acknowledgements. This work was partially supported by grants [OPG 0293307 and OPG 0172633] from the Natural Sciences and Engineering Research Council of Canada (NSERC). This support is gratefully acknowledged. We would also like to thank the logistics manager of our industrial partner for providing us with the relevant data.

---

* Corresponding author: Jacques.Renaud@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec,
Bibliothèque et Archives Canada, 2010

1. Introduction

Automated Storage and Retrieval Systems (AS/RS) are widely used in modern distribution centers (DC) as they provide fast, accurate and efficient material handling on a 24/24 basis. The performance of such automated systems depends greatly on the design but also on several control aspects, such as dwell-point positioning (Bozer & White, 1984; Egbelu & Wu, 1993), work sequencing (Lee & Schaefer, 1996 & 1997; Van den Berg & Gademann, 1999) and storage assignment (Hausman et al., 1976; Graves et al., 1977), which is one of the most important aspects. Roodbergen & Vis (2009) have published an up-to-date survey of the literature on AS/RS.

Storage assignment refers to how the storage locations of an AS/RS should be assigned to products, or product families, in order to optimize the performance of the system. Random Storage (RAN), Full turnover-based Storage (FTB) and Class-based Storage (CB) are well-known policies that are commonly used in practice and are widely discussed in the literature. Among these, the FTB policy consists of assigning storage locations to products based on their individual demand frequency. This policy is closely related to the well-known cube-per-order index (COI) (Heskett, 1963). The FTB policy is widely recognized as the one that minimizes the average travel time of the AS/RS crane, thus maximizing the system throughput (Rosenblatt and Eynan, 1989; Roodbergen and Vis, 2009). Consequently, FTB is currently seen as the policy that leads to the most efficient solution.

This paper studies storage assignment policies for AS/RS and shows that the performance of the FTB policy, compared to RAN or CB storage policies, depends on the modeling assumptions used. When these assumptions are not completely met, like in most of the real settings that we have observed, FTB performance declines rapidly, and so RAN and CB storage may become much more efficient in terms of crane travel times. In particular, this paper presents a situation inspired directly by our industrial partner's AS/RS, demonstrating that the CB storage policy produces much more interesting results. Our results suggest that the FTB policy shouldn't necessarily be considered as the best storage assignment policy compared to the other two policies. Instead, each specific setting should be studied carefully and then the policy to implement should be chosen in terms of the specific setting. This

choice also depends on the number of locations assigned to each product, which may be subject to practical production constraints.

A comprehensive literature review has also led us to conclude that most of the previous research dealing with the previously cited AS/RS policies study them using analytical expressions, which may tend to favor the FTB policy. Rather than examining the system performance from an analytical perspective, we developed a detailed discrete-event simulation model that reproduces the behavior of a unit-load AS/RS. The logic of the simulation model accurately reproduces the behavior of the AS/SR system at our industrial partner's site. Using such a detailed simulation led us to unexpected conclusions about the relative performance of RAN, FTB and CB storage assignment policies.

This paper is organized as follows. Section 2 presents the system under study and its assumptions and reviews the relevant literature. Section 3 provides details about the storage assignment policies used. Section 4 describes the simulation model that we propose. Section 5 reports our empirical results, and section 6 offers our conclusions.

2. Description of the system, its assumptions and a literature review

Hausman et al. (1976) addressed the issue of optimal storage assignment in AS/RS. Their assumptions, which were later used by most of the other authors, are the following:

1. Each pallet holds only one SKU or item type.
2. All storage locations are the same size, as are the pallets themselves.
3. The system consists of a single crane serving a single aisle.
4. Incoming and outgoing pallets are transferred at the same point, denoted the I/O point, which is located at the corner of an aisle.
5. On each side of the aisle is a storage rack with a number of locations equal to R rows and C columns. The crane moves both vertically and horizontally simultaneously, and its vertical and horizontal speeds are such that the time to reach the most distant row from the I/O point equals the time to reach the most distant column (i.e., the system is *square in time* (Bozer and White, 1984)).
6. Interleaving is ignored.

7. The time required by the crane to load or unload a pallet at the I/O point or a storage location is constant, and thus is ignored since it does not influence the performance comparisons.
8. The turnover frequency of each item is known and constant throughout the planning horizon.
9. Studies concern exclusively the average system behavior over the long run.

Under these assumptions, Hausman et al. (1976) proposed an analytical comparison of RAN, CB and FTB assignment policies. For several product demand distributions, they showed that using a two-class CB storage assignment policy led to a distance reduction of up to 50% compared to a RAN policy, while a FTB assignment policy procured distance reductions of up to 70%. They concluded that a significant reduction in crane travel times can be achieved using the FTB storage assignment policy. Schwarz et al. (1978) used a simulation model to validate the analytical results of Hausman et al. (1976). In their model, each pallet entering the AS/RS system is given a specific length of stay, which determines its picking date. An item's length of stay is defined as the inverse of its turnover.

Rosenblatt and Eynan (1989) extended the work of Hausman et al. (1976) which considered only three classes to deal with any number of classes. They used assumptions (1) to (9) and added the following assumption: Items are replenished according to an Economic Order Quantity (EOQ) model. They also assumed that items are ranked according to their contribution to total demand, using an ABC curve. They showed that 91% of the potential FTB gain can be obtained with a four-class AS/RS and 99% of the gain can be obtained with a 12-class AS/RS. Eynan and Rosenblatt (1994) showed that similar results could be obtained with a *rectangular-in-time* AS/RS system.

Kouvelis and Papanicolaou (1995), using the assumptions of Hausman et al. (1976), and Park (2006) developed expected travel time formulas for the dual command AS/RS with two-class storage policies. Kouvelis and Papanicolaou (1995) suggested using their procedure with the procedure proposed by Rosenblatt and Eynan (1989) to obtain good results for a general *n*-class storage policy. Roodbergen and Vis (2009) summarized these results in their review of the literature about AS/RS.

Although not explicitly stated, most of the research mentioned above used a *Location-To-Product Ratio* (LTPR) of 1, which means that only one location is assigned to each product. In theory, an LTPR > 1 can be transformed directly into an LTPR = 1 by making several copies of the product and then dividing the empirical demand uniformly among the copies. It is widely recognized in the literature that, assuming a LTPR=1, (i) compared to CB and RAN storage policies, the FTB policy is the best assignment option according to analytical conclusions; (ii) a CB storage policy is a good alternative to FTB storage assignment policy, and (iii) most of the gain from the FTB policy can also be obtained using from 6 to 12 classes in a CB system. For the remainder of the paper, RAN, CB and FTB notations refer to the number of classes present in the system.

3. Space allocation rules

In this paper, we examine the cases in which LTPR = 1 and LTPR > 1. Clearly, when LTPR > 1, the proportion of storage locations assigned to each product needs to be determined prior to making the storage assignment decisions. As the system considered in this paper has R rows and C columns of single-depth locations on each side of the aisle, the total number of locations is $L = 2RC$. The constant demand rate of each product i is d_i, where $i = 1, \ldots, n$ (see assumption 9). We assume that the items are sorted in decreasing order of the demand (i.e., $d_i \geq d_j$ if $i < j$). The cumulative demand curve can be approximated by the function $G(i)=(i/n)^s$, where $0 < s \leq 1$ (see Hausman et al., 1976; Rosenblatt & Eynan, 1989; Eynan & Rosenblatt, 1994). For example, to obtain a well-known distribution (20/80) in which 80% of the total demand originates in 20% of the products, s must be equal to 0.139.

A commonly used rule-of-thumb consists of assigning locations in proportion to the demand. We found two main variants in the literature. The first one requires calculating the s factor (Hausman et al., 1976). If the first variant is used, the number of locations assigned to each item i, a_i, is

$$a_i = L\left\lfloor i/n - (i-1)/n \right\rfloor^s \qquad (1)$$

The second variant for allocating space uses equation (2) to set a_i:

$$a_i = L\left\lfloor d_i / \sum_{i=1}^{n} d_i \right\rfloor \qquad (2)$$

Many other methods for allocating space can be of interest. Throughout this paper, we will use the first equation with $s = 1.0$ and $s = 0.4$ and also allocate space in proportion to the demand. The following paragraphs show how we implemented the RAN, the CB and the FTB storage assignment policies.

Implementing the RAN, CB and FTB policies

Implementing the RAN policy is quite straightforward: it is only necessary to distribute the required quantity of pallets for each product (a_i) to randomly-chosen locations in the AS/RS system. However, space allocation in class-based (CB) systems is more difficult to implement because which products belong to a class and the number of locations allocated to each class has to be determined simultaneously. As stated above, an analytical approach cannot be used to determine the boundaries of our class-based system. Since the primary objective of this paper is neither to identify the optimal number of classes, nor to identify the optimal size of each class, we allocate space according to the ABC curve. First, products are sorted by the increasing order of their COI. When the number of classes, c, is fixed, the space allocation follows an (ζ_j) scheme, where $j = 1, \ldots, c$ and where $\sum_j \zeta_j = 1$, meaning that ζ_j% of the products are assigned to class j. Then, the number of locations assigned to a specific product is determined by using equation (1) for a given s, or by using equation (2).

If $c = 2$, a possible class-division implementation is (ζ_1; ζ_2) = (0.2; 0.8), meaning that the first 20% of the products with the largest COI are assigned to the first class and the last 80% of the products are assigned to the second class. If C_j is the set of products assigned to class j, the total number of locations of class j will be $l_j = \sum_{i \in C_j} a_i$. Then, the l_1 best locations (i.e., the closest ones to the I/O point) correspond to class 1. The l_2 other locations correspond to class 2, and so on. Within a given class, the products are randomly assigned to the locations of this class. A system with one class corresponds to a random (RAN) storage policy. A system with two classes corresponds to a class-based (CB) storage policy.

However, if the number of classes equals the number of products (i.e., $c = n$), the system is a full turnover-based (FTB) system.

4. Simulation Model

In order to perform our experiments, we developed an accurate simulation model based on a discrete-event engine. The code was implemented in VB.NET. One of the reasons for this language choice is the great flexibility offered by an object-oriented language. It allowed us to test a wide variety of scenarios just by varying the objects' properties. The model was developed to reproduce the storage operations at our industrial partner's main facility, which is located in Quebec City, Canada. The following sections describe the model components in more detail.

Demand Generator (Input modeling)

In order to manage the large quantity of historical data, we used MS Access, a database management software. Our industrial partner collected data about all movements of 75 high-volume items handled by the AS/RS. For each product, we modeled as discrete probability distributions the *order frequency*, which gives the probability that the product would appear in a given customer order, and the *average order quantity*, which gives the most probable number of pallets required if the product appears in an order. The turnover rate of each product is the product of these two parameters.

To model these two distributions, more than 1000 orders from 6 months of system activity were analyzed by a commercial statistical tool, the *Input Analyser*, included in *Arena*, the simulation software from Rockwell Automation. The *Input Analyser* was used to compare variates from prescribed probability distributions for the data provided by our partner. It was also used to choose the distribution, and its parameters, that best fit the data sample. To this end, *Input Analyser* used maximum likelihood estimation to estimate the parameters of several distributions commonly found in simulations. Then, the possible distributions were sampled and the variates were compared to the data. Statistical goodness-of-fit tests, including Chi-Square and Kolmogorov-Smirnov tests, were then run automatically, and the best-fitting distribution was selected. Once the distributions for *order frequency* and *average*

order quantity had been selected for each product, the demand could be simulated by sampling these distributions.

Crane simulator

The AS/RS system was modeled as a matrix of storage locations, with a crane that moves between the I/O point and the locations according to the task to be accomplished. The system is a two-sided storage rack aisle, which is served by a single crane. (However, our simulation model can represent other configurations.) The aisle is C slots long and R slots high for a total of $2CR$ single-depth racks. The AS/RS crane moves simultaneously on both axis (Chebyshev metric) at an average speed of 0.4 m/s vertically and 1 m/s horizontally. The crane can only handle a single pallet at a time. In our experiments, the pickup and deposit times were assumed constant at 10 seconds, and the crane's acceleration was assumed to be instantaneous.

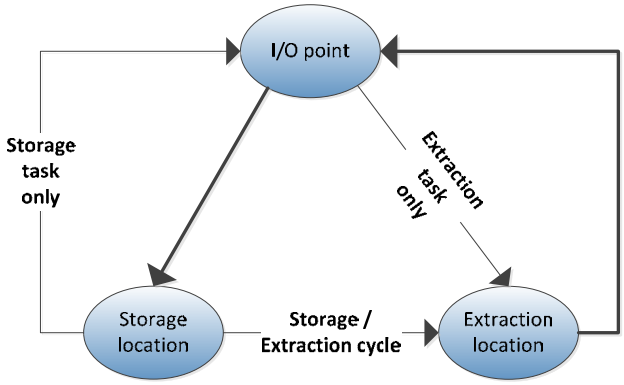


Figure 1: Model of the crane movements

Figure 1 shows the possible moves that can be performed by the crane, the bold lines indicating legs where the crane is loaded with a pallet. From the dwell (start) point, the crane can execute a storage task or an extraction task. A storage task starts at the I/O point and ends at the storage location. Obviously, if two storage tasks are consecutive, the crane needs to travel from the storage location of the first task to the I/O point prior starting the second task. Extraction tasks start by moving the crane from its current position (either the I/O point or the location were the last storage was done) to a retrieval location and end the cycle at the I/O point. In order to minimize the total distance, tasks can be combined into

storage/extraction cycles instead of using single tasks. A cycle starts with the crane at the I/O point. A pallet is loaded, and the crane leaves for the storage location. Once there, the pallet is unloaded, and the crane moves to the extraction location. At the extraction location, the pallet is loaded, and the crane moves back to the I/O point where the pallet is unloaded. This cycle consists of only three "legs" instead of the four legs required to perform two independent operations.

Despite its apparent simplicity, the performance of the crane accomplishing the storage and extraction tasks is highly dependent on the strategies used to manage storage assignment. The choices about the storage location of a pallet that needs to be stored and the extraction location from which a pallet is extracted are particularly important. These choices are described in the following two paragraphs.

The storage location for the arriving pallet

We considered two rules to guide a pallet arriving in the system to the location where it will be stored: a *random location* (RAND) or the *closest open location* (COL). The RAND rule identifies all the available locations that could receive the arriving pallet and selects a location randomly (i.e., all the locations have the same chance to be selected). The COL rule also identifies all the available locations, but it selects the location that is closest to the I/O point.

The location from which the required pallet is extracted

When the crane processes a product extraction request, it could be that only a single pallet of this particular product is stored in the AS/RS, in which case the crane has no choice but to extract it. However, it could also be that the system holds several pallets of this product at different locations. If this is the case, two rules can be used to choose from which location the pallet will be extracted: a *random location* (RAND) or the *shortest leg location* (SL). The RAND rule selects the location randomly from the possible locations. The SL rule selects the location requiring the shortest trip from the current crane position. For a single extraction task, the SL rule selects the pallet closest to the I/O point; for a Storage/Extraction

cycle, the SL rule selects the pallet closest to the location where the storage task was accomplished.

Therefore, the crane simulator decides where to store the pallets and where to extract pallets depending on the storage and retrieval requests. The list of requests and the order in which they are treated is managed by the task dispatcher.

Task Dispatcher

In the model described above, storage and extraction requests are coordinated by the *Task Dispatcher (TD)*. Sequencing storage and extraction requests is an important issue in AS/RS management as demonstrated by Chen et al. (2010) which minimize interleaving time. We refer interested readers to Roodbergen and Vis (2009) for their review of the literature about sequencing storage and retrieval requests. The particularities of the system being studied require that the task dispatcher manage the demand (i.e., orders to be extracted) and the replenishment flow in different manners. Although the system provides reasonable freedom when choosing which pallet to extract, storage requests are merely a sequence of pallets arriving on a single conveyor from the production area to the I/O point of the AS/RS system. The system mechanics do not allow an arbitrary selection of a pallet, but only allows the first pallet in the line to be selected. Therefore, pallets are loaded on the crane in a "first in, first out" (FIFO) manner.

Extraction scheduling offers much more flexibility to the task dispatcher. Indeed, demand arrives in the system as “orders”, each containing a list of products to be picked. The sequence in which the products are picked within a given order is not important, provided that the orders are handled in a FIFO manner. Although the storage request sequence cannot be modified, the sequence is known in advance. Therefore, the task dispatcher’s goal is to synchronize extractions with a given storage list in order to minimize the total distance travelled by the crane.

In all the experiments reported in this paper, the dispatcher was configured to deal with the requests in the sequence that they appear in the order. The reason for this configuration is that we aim to accurately assess the impact of the space allocation rules, and having an

intelligent dispatcher could deeply perturb our observations. However, even in its simplest configuration, the dispatcher tries to avoid operational delays due to product stockouts. Thus, whenever a product is unavailable in the AS/RS system, the request is delayed until a pallet of such a product enters the system.

Manufacturing simulator

In our model, extractions are generated, or simulated, using probability distribution sampling. However, it is possible to model several alternatives (e.g., push, lots, pull) and then simulate system replenishment. We therefore imagined a pull system with a single production line linked to the AS/RS system. As soon as a pallet was extracted from its storage location, a production request of size q was sent to the production line. The production time t_i for a unit of each product i was known, and setup times could be adjusted by the user. In our experiments, we chose to fully synchronize replenishments (production) to the demand (extractions). Thus, requests were treated in a FIFO manner, and q was set to 1. Finally, we assumed a constant production time for all the products. The production rate chosen was faster than the AS/RS operations. Consequently, although no congestion or waiting occurred on the production line, the pallets to be stored could accumulate at the I/O point.

Discrete-Event Simulation Engine

The simulation engine is similar to the one proposed by Pidd (1995), consisting of a three-phase algorithm that allows the clock to be advanced asynchronously from one event to the next. The simulation engine works with a list of events sorted by their execution time. Each time an event is executed, the system state is modified accordingly, and the simulation clock moves to the following event in the list. Sometimes, the execution of an event does not change the state of the system, but rather generates new events to be added to the list. In this case, the time at which the event will be executed is computed or simulated.

Events are classified into bounded (B) and conditional (C). B-type events are those for which the execution date can be predicted (or simulated) by the system. For example, let t be the current time at which a particular event (“travel from I/O to storage location”) is

executed. Execution of this event implies the generation of a new event ("unload the pallet at the storage location") whose execution date can be stated as $t + d$, where d is the travelling time between the I/O point and the selected location. If travelling times are modeled using probabilistic distributions, d may be a deterministic value or a variate. On the other hand, the execution date for conditional events cannot be determined in advance because they depend on a given system state. For example, it is not possible to set the execution date of an event like "travel from I/O to storage location" in advance because the event will be executed as soon as the following two conditions are satisfied: (1) a storage request is waiting to be processed in the queue and (2) the crane is stopped at the I/O point.

Clearly, type-B events are in the events list, but type-C events depend on how the system state evolves. Since the system state is only changed after an event has been executed, it follows that the algorithm needs to check the conditions for type-C events only after a B-type event has been executed. Therefore, the approach is very efficient, as it minimizes the computation time because calculations are only done when an event has already occurred.

Tables 1 provide the entities (the crane, storage and extraction request lists, and a system variable keeping trace of the last activity performed) and their possible states. Table 2 lists the system events and gives, for each event, its type, the conditions allowing its execution (for type-C events), the changes in the system state associated to its execution and, if a new event is generated, the generated event. Our model was conceived for the case of one crane and one aisle, but it can be extended to several aisles in a straightforward manner made possible by the object-oriented modeling approach employed.

Entity	**State**
Crane	• Idle • In Movement • Loading • Unloading
Last activity	• Extraction • Storage
Storage queue	• number of requests waiting
Extraction queue	• number of requests waiting

Table 1: Entities and states

Event	Type	Execution Condition	System State Changes	New event generated
Start storage task	**C**	Crane is idle Storage queue > 0 (Last activity = extraction) OR (Last activity = storage AND Extraction queue = 0)	Crane = In movement (to I/O)	Arrival at I/O for storage task
Arrival at I/O for storage task	B		Crane = Loading	Move to storage location
Move to storage location	B		Crane = In movement (to storage location)	Arrival at storage Location
Arrival at storage location	B		Crane = Unloading	Storage completed
Storage completed	B		Crane = Idle Storage queue = Storage queue – 1 Last activity = storage	
Start extraction task	**C**	Crane is idle Extraction queue > 0 (Last activity = storage) OR (Last activity = extraction AND Storage queue = 0)	Crane = In movement (to extraction location)	Arrival at extraction location
Arrival at extraction location	B		Crane = Loading	Move to I/O
Move to I/O	B		Crane = In movement (to I/O)	Arrival at I/O
Arrival at I/O	B		Crane = Unloading	Extraction completed
Extraction completed	B		Crane = Idle Extraction queue = Extraction queue – 1 Last activity = extraction	

Table 2: Simulation engine events and the logic behind these events

Our three-phase algorithm works as follows. In phase A, it seeks the first event in the sorted list and the system clock is set to the execution time of the current event. In phase B, the current event is executed (i.e., the required changes to the system are applied and, if

necessary, new events are added to the list). In Phase C, it verifies whether or not the system state makes it possible to execute type-C events by testing the necessary conditions given in Table 2. After executing each A-B-C loop, the algorithm verifies whether or not the simulation stop conditions have been met. If so, the algorithm stops; if not, it moves back to Phase A.

A simple example is used to illustrate the algorithm's execution process. Let us assume that non-empty lists of extraction and storage requests produced by the dispatcher are available at time $t = 0$, and that the event list is empty. An initialization process resets the system clock and places the crane, in an idle state, at the I/O point. Without loss of generality, the initialization process also sets the crane's last activity to "storage" so it always starts with an extraction task at the beginning of a simulation run. Since the event list is empty, the algorithm skips Phase A and Phase B and moves directly to phase C. The type-C conditions that need to be tested are given in Table 2.

For instance, the event "Start extraction task" is created if the crane is idle and if there is at least one extraction request in the queue. When this event is executed, the crane's state changes to "In movement" and a new event, "Arrival at extraction location", is added to the events list. The execution time for this new event is the current clock time plus the expected travelling time, which can be computed based on the distance between the I/O point and the targeted extraction location chosen according to the RAND or COL rules. Since the crane's state is "In movement", none of the other type-C events can be executed. Thus, the algorithm moves back to phase A. The next event in the list is the "Arrival at extraction location", which has just been generated. The algorithm selects this event, updates the clock to its execution time and then executes the event, which changes the crane's state to "Loading" and creates the "Move to I/O" event. The current event can then be deleted from the events list. The algorithm moves on to Phase C but cannot meet the type-C conditions. In the same manner, all events are executed until the extraction request is completed, changing the state of the crane to "Idle".

5. Empirical Results

The goal of our experiments was to measure the performance of the storage assignment policies (i.e., RAN, CB and FTB) combined with crane movement rules (RAND, COL, SL). To this end, five instances, each with 2000 extractions, were generated using the input modeling approach presented above. The system performance, in terms of total distance travelled for the crane to perform the extraction tasks (and the storage tasks) associated to each instance, was collected using the simulation model presented in the previous section. All the configurations to be tested used the same task dispatcher so, for each instance, the extraction list is the same and is independent of the tested configuration. In this way, we lessen the variance reduction problems associated to simulation experiments (for example, see Law and Kelton, 2000).

For each instance, the AS/RS was fully replenished at simulation time t=0 in a pseudo-random manner. In order to give the system some slack, extractions were not compensated via replenishment until 20% of the storage space in the AS/RS was free. After that, every extraction triggered a manufacturing request. Tests were run for a single-side single-depth AS/RS that was C=25 columns long and R=12 rows high for a total of 300 locations. Although our simulation tool is flexible (i.e., simulation multi-aisle settings or intelligent dispatchers), we chose to run our tests on the simplest configuration in order to better isolate the effects of storage assignment policies.

We divided our analysis into three parts. First, in Section 5.1, we provide the computational results for a system with L = 300 locations and n = 300 products, which had a Locations-To-Product Ratio (LTPR) of 1. We hypothesized that the results in this first part would reproduce exactly the usual results published in the literature. Since the results in this first part reproduced exactly the usual results published in the literature, we used this first set of experiments to validate the behavior of our simulation model. Second, in Section 5.2, we provide the computational results for a system with L = 300 locations and n = 75 products, which had a LTPR of 4. For this case, the literature results do not always hold true, as simple class-based (CB) assignment policy can perform much better than the full turnover-based (FTB) policy. The experiments run in Sections 5.1 and 5.2 consider three storage assignment cases (one class, two classes and n classes), and for each case, two crane

movement rules (RAND and COL/SL). Finally, in Section 5.3, we explain why RAN and CB policies may perform better than an FTB policy, by looking at the details of system behavior.

5.1 Results for 300 locations and 300 products (LTPR=1)

For these "validation" tests, we generated a single long instance with 4000 extractions for 300 products, where the first 20% of the fast mover products represented 52% of the total demand. In this case, the approximate ABC curve was best represented with $s = 0.40$. Clearly, since LTPR=1 each product had a single location in the system. In this context, class size only depends on the number of products it contains. When a two-class system was designed, the first class received 20% of the products (ζ_1=0.20, ζ_2=0.80) When we used a three-class system, the first class received the first 20% of the products, the second class the next 20% and the third class the remaining 60% (ζ_1=0.20, ζ_2=0.20, ζ_3=0.60). Products were initially randomly positioned within each class. For the FTB system, products were sorted in decreasing order of their order frequency and iteratively assigned to the location closest to the IO point. Figure 2 presents the product order frequency and the corresponding ABC curve.

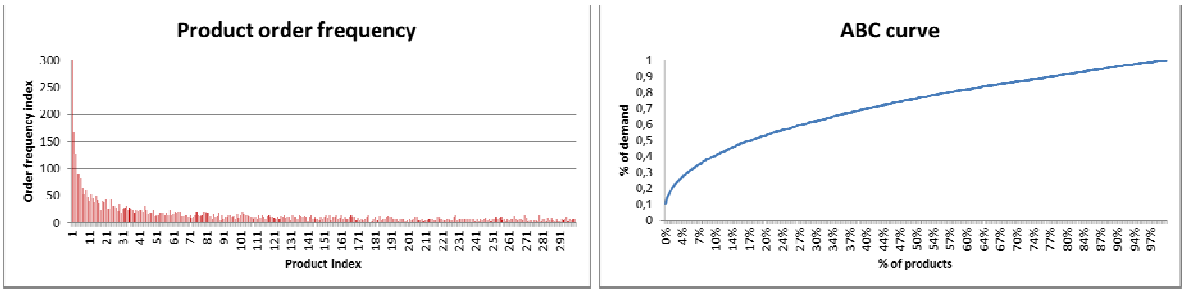


Figure 2: Distribution characteristics for n = 300 products

Table 3 and Figure 3 show the results if the AS/RS is divided into 1, 2, 3, and n classes in terms of two crane movement rules: RAND/RAND (columns R/R for concision) and *closest open location–shortest leg* (columns COL/SL). Since each product occupies only one location, the random and the shortest-leg extraction rules lead to the same result for the FTB implementation. Clearly, the system performance improves with the number of classes, with the best results being obtained with n classes. Within the same storage assignment

configuration (number of classes), results produced by the COL/SL crane movement rule outperforms the RAND rule.

	1 class (RAN)		**2 classes (CB)**		**3 classes (CB)**		***n* classes (FTB)**	
	R/R	COL/SL	R/R	COL/SL	R/R	COL/SL	R/R	COL/SL
Instance1	3 770	3 537	2 837	2 812	2 784	2 758	2 557	2 557

Table 3: Total crane travel times for the case where LTPR = 1 (in minutes)

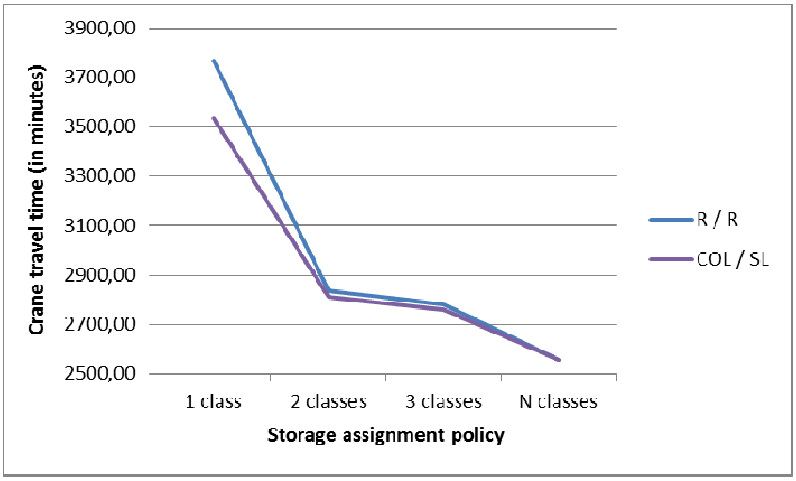


Figure 3: Crane travel time (in minutes) for different crane movement rules and storage assignment policies

5.2 Results with 300 locations and 75 products (LTPR = 4)

In this section, we report our results for a system that has 300 locations but only 75 products. For this case, we generated five 2000-extraction instances using the demand generator presented in Section 4. We did not consider the 3-class case because its behavior was expected to be somewhere between the 2-class and the *n*-class cases. The empirical ABC curve was best approximated with $s = 0.40$. The first 20% of the fast mover products represented 52% of the total demand. Figure 4 presents the product order frequency and the corresponding ABC curve.

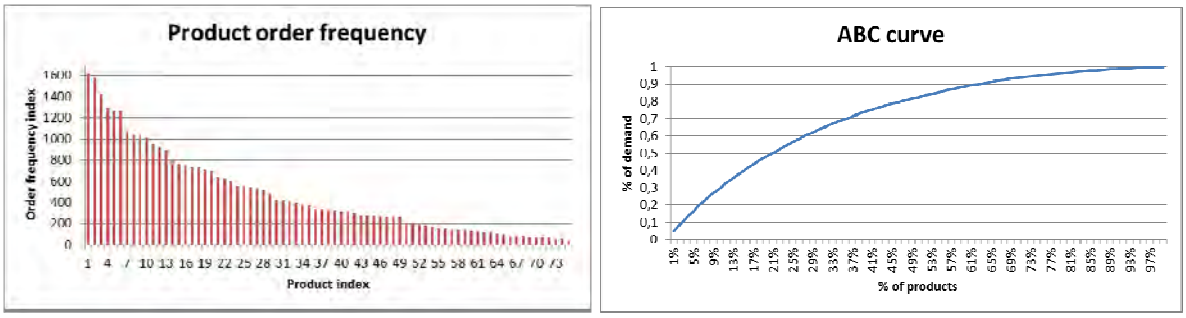


Figure 4: Distribution characteristics for n = 75 products

Since the number of locations is greater than the number of products, we had to determine the number of locations allocated to each product. As mentioned above (section 4), this allocation can be done using equation (1) with different s values. Thus, we decided to use two different values of parameter s and to calculate the allocations with these two values.

Results with s = 1

First, we began with s = 1, which corresponds to a balanced allocation in which each product receives exactly 4 locations. When we used two classes, the first 20% of the fast mover products were assigned to the first class, with 20% of the available space. Results for the total crane travelling time required to perform the 2000 extraction tasks and the associated storage tasks are summarized in Table 4. Each row in the table corresponds to a different instance. The columns R/R refer to the crane movement rule using RAND/RAND selection of storage and extraction locations. The columns COL/SL refers to the crane movement rule using the closest-open-location rule to select the storage locations and the shortest-leg rule to select extraction locations.

	1 class (RAN)		**2 classes (CB)**		***n* classes (FTB)**	
	R/R	COL/SL	R/R	COL/SL	R/R	COL/SL
Instance1	1 522.4	1 222.9	1 276.3	1 080.3	1 119.8	1 123.3
Instance2	1 584.1	1 280.5	1 272.9	1 120.6	1 131.9	1 130.5
Instance3	1 618.7	1 266.0	1 373.2	1 185.0	1 195.6	1 192.4
Instance4	1 604.1	1 377.3	1 331.8	1 161.3	1 167.2	1 164.7
Instance5	1 539.4	1 277.8	1 284.3	1 121.4	1 104.3	1 101.4
Average	**1 573.7**	**1 284.9**	**1 307.7**	**1 133.7**	**1 143.7**	**1 142.5**

Table 4: Total crane travel times (in minutes), where s = 1

Table 4 shows that the expected configuration performance generally increases with the number of classes. Also, within a class, the crane movement rule COL/SL always produced better results than RAND/RAND. However, unexpected is the fact that the COL/SL result is better with two classes than with *n* classes.

The average results between the 2-classes and *n*-classes when using the COL/SL crane movement rule are so close (1 133.7 and 1 142.5, respectively) that we performed two paired-mean-sample tests to verify whether or not the results produced by the two configurations were statistically different. To this end, we calculated the mean and a confidence interval on the random variable Y_i = COL/SL(*n-classes*)$_i$ - COL/SL(*2-classes*)$_i$, using the five instances mentioned above. We obtained $\hat{Y}$ = 8.75 minutes and a 95% confidence interval (or half-width) of ± 0,75 minute. This extremely low variability value (i.e., a very small half-width) suggested that no more samples would be needed to make a statistically significant decision. Since $\hat{Y}$ is positive, and the confidence interval around $\hat{Y}$ does not contain 0, we can conclude with a 95% confidence level that under the COL/SL movement rule, the results produced by the 2-classes and *n*-classes configurations are different and that the 2-classes configuration is in fact better that the *n*-classes. According to these results, we can expect an average relative improvement of 100*(8.75/1142.5) ~ 0.76%.

Results with s = 0.4

We performed another set of tests using the same 5 instances that we used with s = 1, but, this time, the location allocation was done with s = 0.4, which means that the number of locations assigned to each product reflects its importance with respect to the total demand. Results for the total crane travelling time required to perform the 2000 extraction tasks and the associated storage tasks are summarized in Table 5. Again, each row in the table corresponds to a single instance and the columns R/R and COL/SL refer to the two crane movement rules already described above.

	1 class (RAN)		2 classes (CB)		N classes (FTB)	
	R/R	COL/SL	R/R	COL/SL	R/R	COL/SL
Instance1	1 506.4	1 101.1	1 410.3	1 193.6	1 331.1	1 324.2
Instance2	1 588.4	1 143.9	1 452.4	1 217.3	1 357.7	1 342.7
Instance3	1 669.4	1 242.3	1 581.2	1 300.4	1 447.8	1 437.6
Instance4	1 677.3	1 179.9	1 527.7	1 289.3	1 411.7	1 402.2
Instance5	1 554.8	1 100.6	1 415.7	1 226.7	1 320.4	1 311.6
Average	**1 599.3**	**1 153.6**	**1 477.5**	**1 245.4**	**1 373.7**	**1 363.7**

Table 5: Total crane travel times (in minutes), where $s = 0.4$

Again, when using the RAND/RAND rule to manage the crane movement, the system performance improves with the number of classes. We also observe that, as in the previous experiments, the results produced by the COL/SL crane movement rule outperforms the ones produced by the random rule, independently of the number of classes considered. However, it was quite unexpected that, when the crane is managed with the COL/SL rule, the system performance decreases as the number of classes increases, with the best result being obtained with the 1-class setting. These results show that the system performance in terms of storage assignment policies heavily depends on the number of locations assigned to each product. This case presents a counter-intuitive situation in which a single class storage assignment leads to better results than the FTB policy. In order to statistically assess the difference between the best configuration promoted by the literature, i.e. FTB storage assignment with COL/SL crane movement rule, and the best configuration suggested by our experiments (1-class, COL/SL), we used again paired-mean-sample tests, with Y_i = COL/SL(*n-classes*)$_i$ - COL/SL(*1-class*)$_i$ yielding a $\hat{Y}$ = 210.1 minutes and a 95% confidence interval (or half-width) of ± 0.43 minute. According to these results, we can expect an average relative improvement of 100*(210/1 363.7) ~ 15.4%.

5.3 Performance explanation

This section tries to explain why the results reported in sections 5.2 differ from the expected behavior according to the literature. Examining our results and the results of previously published papers on AS/RS led us to conclusion that the underlying hypothesis, which holds that each location of a given product has the same probability of being visited, plays a crucial role in the behavior that is expected. This hypothesis implies that, within a class,

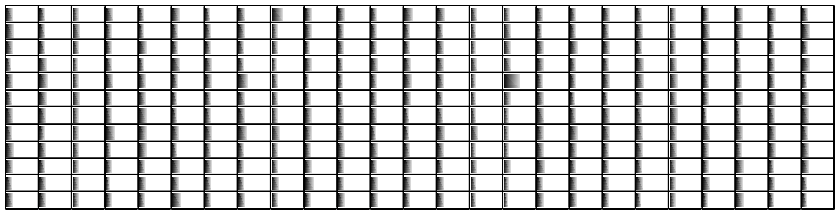
each location has the same probability of being visited. In this case, the average travel distance within a class is equal to the distance to its center point. Clearly, this hypothesis is only satisfied when LTPR=1. For this case, our results confirm both analytical and simulation results previously published, which hold that the system performance increases with the number of classes, with the best performance being obtained by FTB (see Table 3).

To better illustrate our point, let us look at Figure 5, which reports the number of times that each location was visited in the AS/RS under study. In Figure 5, a black bar within each location gives its relative number of visits; the higher the number of visits, the longer the bar. Figure 5a shows the results for 300 locations, 300 products and a single class, with RAND/RAND crane movement. As expected, Figure 5a illustrates that the use of the various locations is mostly uniform. Figure 5b illustrates the FTB solution for an LTPR = 1, in which the most intensively used locations, located at the left top corner, are closer to the I/O point.

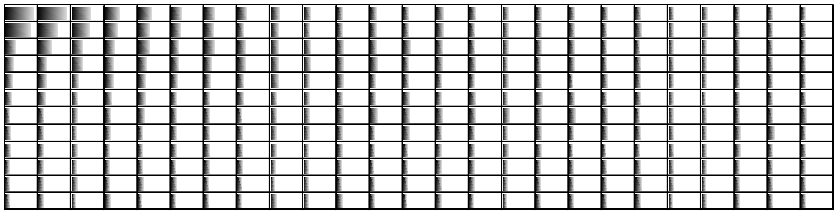
When products have more than one location, the crane movement rule is of paramount importance. Figures 5c and 5d show the results for 75 products, 300 locations and s = 1.0, where RAND/RAND is used as crane movement rules. When RAND/RAND is used, locations are equally visited within a class (Figure 5c). In comparison, when COL/SL is used, the visits are not so uniformly distributed (Figure 5e). We showed that when products have more than one location, the COL/SL rule easily outperformed the RAND/RAND rule. This behavior can be explained by the fact that locations are no longer uniformly visited, as supposed by analytical models, which reduces the average travel time. Figure 5e shows the relative number of visits to each location in the 1-class system using the COL/SL rule and s = 1.0. Although the products are initially randomly distributed throughout the AR/RS, their locations are clearly rearranged over time so that fast mover items move closer to the I/O. In this case, the best locations are used heavily, and this concentrated usage becomes higher than the usage with the FTB policy. In this situation, the FTB policy restricts the system adaptations to the orders, which is done automatically with only one class. As Figure 5f shows, when products share multiple locations, the system behaves differently from the expected situation with one location per product (Figure 5b). In this case, it is easy to see that, even if each product has 4 dedicated locations, most of the picks are concentrated in the

nearest one, which helps to explain why the configuration shown in Figure 5f is not as smooth as the one shown in 5a.

Finally, Figure 5g presents a 2-class system with s=1.0 and RAND/RAND crane movement rules in which the first class frontier is outlined. For this configuration, the average total travel time is 1 307.7 minutes. In this case, the expected behavior is obtained, and the locations are almost uniformly visited within the class. In this situation, it is correct to assume that the average distance travelled to store a product within a class is closer to the center point of the class. However, when the crane movement is governed by the COL/SL rule and LTPR > 1, the movements within the first class are clearly concentrated closer to the I/O point, as clearly shown by Figure 5h. In this case, the average total time traveled is much lower (1 133.7 minutes) since the average distance moved is much lower than the distance to the class center point. This explains why a single class system can achieve a much better performance than predicted by analytical models and performs better than FTB.

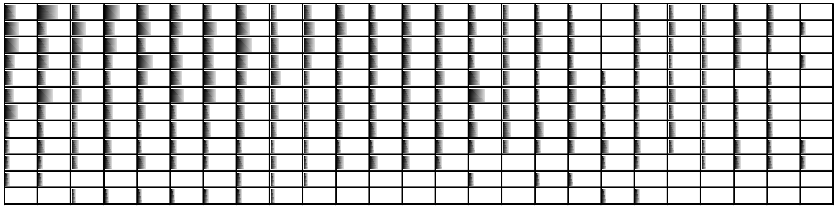


a) 300 products, 1 class, crane movement (RAND/RAND)

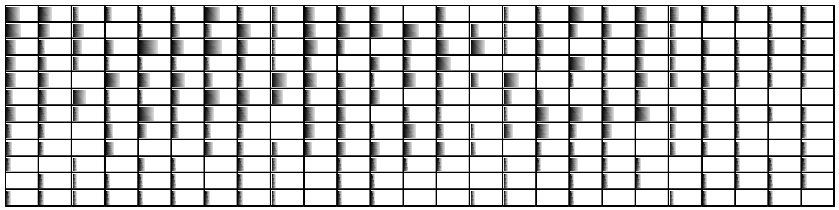


b) 300 products, FTB, crane movement (RAND/RAND)

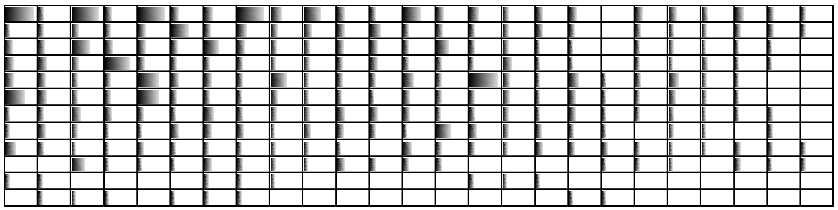
c) 75 products, 1 class, *s*=1.0, crane movement (RAND/RAND)



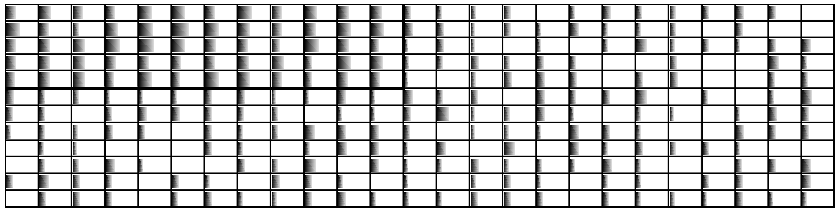
d) 75 products, FTB, s=1.0, crane movement (RAND/RAND)



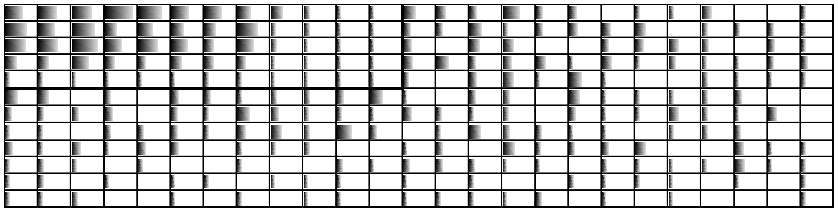
e) 75 products, 1 class, *s*=1.0, crane movement (COL/SL)



f) 75 products, FTB, *s*=1.0, crane movement (COL/SL)



g) 75 products, 2 classes, *s*=1.0, crane movement (RAND/RAND)



h) 75 products, 2 classes, *s*=1.0, crane movement (COL/SL)

Figure 5: Extraction frequencies of the 300 AS/RS locations

6. Conclusion

This paper studies storage assignment policies for AS/RS and shows that, compared to random (RAN) storage or class-based (CB) storage policies, the performance of the Full Turnover-based (FTB) policy depends on the modeling assumptions used. When these assumptions are not completely met, as in most of the real settings that we have observed, the performance of an FTB system declines rapidly and other simple assignment policies, such as RAN or CB policies, may become much more efficient in terms of crane travel times. In particular, this paper presents a simulation model that accurately reproduces our industrial partner's unit-load AR/RS. Our simulations in this industrial context demonstrated that, unlike the behavior expected according to the theoretical results in the literature, a single class storage assignment policy produces much more interesting results, yielding crane travel-times of up to 19% less than the ones obtained using the FTB policy. Therefore, our results suggest that the FTB policy shouldn't necessarily be considered as the best storage assignment policy in absolute terms. Instead, each specific setting needs to be carefully studied and then the policy to implement should be chosen according to the specific setting.

Acknowledgement

This work was partially supported by grants [OPG 0293307 and OPG 0172633] from the *Natural Sciences and Engineering Research Council of Canada* (NSERC). This support is gratefully acknowledged. We would also like to thank the logistics manager of our industrial partner for providing us with the relevant data.

References

Bozer Y.A. & White J.A., 1984, Travel-time models for automated storage/retrieval systems. *IIE Transactions*, **16**, 329-338.

Chen L., Langevin A. & Riopel D., The storage location assignment and interleaving problem in an automated storage/retrieval system with shared storage, *International Journal of Production Research*, **48**, 2010, 991-1011.

Egbelu P.J. & Wu C.T., 1993, A comparison of dwell point rules in an automated storage/retrieval systems. *International Journal of Production Research*, **31**, 2515-2530.

Eynan A. & Rosenblatt M. J., 1994, Establishing zones in single-command class-based rectangular AS/RS. *IIE Transactions*, **26**, 38-46.

Graves S.C., Hausman W.H., & Schwarz L.B., 1977, Storage-retrieval interleaving in automatic warehousing systems. *Management Science*, **23**, 935-945.

Hausman W.H., Schwarz L.B., & Graves S.C., 1976, Optimal storage assignment in automatic warehousing systems. *Management Science*, **22**, 629-638.

Heskett, J. L., 1963, Cube-per-order index – a key to warehouse stock location. *Transportation and Distribution Management*, **3**, 27-31.

Kouvelis P., & Papanicolaou V., 1995, Expected travel time and optimal boundary formulas for a two-class-based automated storage/retrieval system. *International Journal of Production Research*, **33**, 2889-2905.

Law, A.M. & Kelton, W.D., 2000, Simulation Modeling and Analysis. 3rd Edition, Mc Graw Hill Higher Education.

Lee H.F. & Schaefer S.K., 1996, Retrieval sequencing for unit-load automated storage and retrieval systems with multiple openings. *International Journal of Production Research*, **34**, 2934-2962.

Lee H.F. &Schaefer S.K., 1997, Sequencing methods for automated storage and retrieval systems with dedicated storage. *Computers & Industrial Engineering*, **32**, 351-362.

Park B. C., Performance of automated storage/retrieval systems with non-square-in-time racks and two-class storage. *International Journal of Production Research*, **44**, 2006, 1107-1123.

Pidd M., 1995, Object-orientation, Discrete simulation and the Three-Phase approach. *The Journal of the Operational Research Society*. **46**, , 362-374.

Roodbergen K. J. & Vis I. F.A., 2009, A survey of literature on automated storage and retrieval systems. *European Journal of Operational Research*, **194**, 343-362.

Rosenblatt M. J. & Eynan A., 1989, Deriving the optimal boundaries for class-based automatic storage/retrieval system. *Management Science*, **35**, 1519-1524.

Schwarz L.B., Graves S.C. & Hausman W.H., 1978, Scheduling policies for automatic warehousing systems: Simulation results. *AIIE Transactions*, **10**, 260-270.

Van den Berg J.P. & Gademann A.J.R.M., 1999, Optimal routing in an automated storage/retrieval system with dedicated storage. *IIE Transactions*, **31**, 407-415.